REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
OCTOBER 2014	TECHNICAL PAPER	JAN 2013 – JAN 2014

4. TITLE AND SUBTITLE

SECURITY: A KILLER APP FOR SDN? (POST PRINT)

5a. CONTRACT NUMBER
FA8750-13-2-0023

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
62303E

6. AUTHOR(S)

Dongting Yu, Andrew Moore, Chris Hall, Ross Anderson

5d. PROJECT NUMBER
PROC

5e. TASK NUMBER
ED

5f. WORK UNIT NUMBER
UI

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Indiana University
901 E. 10th St.
Bloomington, IN 47401

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/
Information Directorate
Rome Research Site/RITA
525 Brooks Road
Rome NY 13441-4505

Defense Advanced Research
Projects Agency (DARPA)
675 North Randolph St
Arlington, VA 22203

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TP-2014-048

12. DISTRIBUTION AVAILABILITY STATEMENT
Distribution Approved For Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. SUPPLEMENTARY NOTES
This work was funded in whole or in part by Department of the Air Force contract number FA8750-13-2-0023.

14. ABSTRACT
Software Defined Networking (SDN) has been developed rapidly and is now used by early adopters such as data centres. It offers immediate capital cost savings by replacing proprietary routers with commodity switches and controllers; the use of computer science abstractions in network management offers operational cost savings, with performance and functionality improvements too. However, there is a third class of benefits, that will come into their own once SDN is deployed in less controlled environments: and that is security. Traditional network architectures have many points of serious failure; the compromise of a single router can lead to significant attacks. SDN enables network designers to limit the damage that compromised switches can do, and thus can support more resilient and survivable networks in environments where opponents may have access to some of the infrastructure. In this paper we discuss the security aspects of SDN, and the possible opportunities that result.

15. SUBJECT TERMS
Software Defined Network, SDN, Network Routing, Security

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			CARL THOMAS
U	U	U	UU	7	19b. TELEPHONE NUMBER (*Include area code*) N/A

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

Security: a killer app for SDN?

Dongting Yu
Dongting.Yu@cl.cam.ac.uk
University of Cambridge

Andrew W. Moore
Andrew.Moore@cl.cam.ac.uk
University of Cambridge

Chris Hall
Chris.Hall@highwayman.com
Highwayman Associates Ltd.

Ross Anderson
Ross.Anderson@cl.cam.ac.uk
University of Cambridge

ABSTRACT

Software Defined Networking (SDN) has been developed rapidly and is now used by early adopters such as data centres. It offers immediate capital cost savings by replacing proprietary routers with commodity switches and controllers; the use of computer science abstractions in network management offers operational cost savings, with performance and functionality improvements too. However, there is a third class of benefits, that will come into their own once SDN is deployed in less controlled environments: and that is security. Traditional network architectures have many points of serious failure; the compromise of a single router can lead to significant attacks. SDN enables network designers to limit the damage that compromised switches can do, and thus can support more resilient and survivable networks in environments where opponents may have access to some of the infrastructure. In this paper we discuss the security aspects of SDN, and the possible opportunities that result.

1. INTRODUCTION

One question facing advocates of software defined networking is: what can it do that we cannot already do with existing technology? The move from proprietary router technologies to commodity switches and controllers holds out the promise of significant cost savings, and has led to SDN deployment in data centres where these cost savings are sufficient to compensate for the added engineering effort and technology risk. In the medium term, the programmability of the controllers opens up the prospect of much better network management, as we move from large scripts and proprietary command lines to proper computer science abstractions.

In this position paper, we argue that there is a further significant opportunity, in that SDN networks could be secured much better, particularly against attacks resulting from local compromise of switches. This will become increasingly more important as networks get larger and serve more heterogeneous tenants.

Consider the following possible deployment scenarios:

1. a data centre with 100,000 machines, connected via commodity switches and a hierarchy of top-of-rack, end-of-rack and facility-wide controllers;
2. a large tier-1 transit provider with many hundreds of Points of Presence (PoPs) in 100 countries;
3. a system services company that provides private networks to 100 banks spread across 500 buildings mostly in New York, London, Frankfurt, Tokyo and Hong Kong;
4. a large international airport with 50,000,000 passengers a year, served by 100,000 staff who work for 1,000 organisations. These include not just government agencies dealing with crime, immigration and intelligence, but hundreds of airlines and freight companies directly competing with each other.

The first of these is the typical early adopter scenario, and the next three are likely future adopters over the next 5–10 years as the technology matures. Going down the list, two things happen: the technological sophistication (in terms of the number of PhD-level network engineers employed) decreases, and the level of physical control over network hardware decreases. Large ASes have a lot of equipment on premises they control, but inevitably have routers at far-flung points of presence. Large heterogeneous commercial networks may also have customers that are vigorous competitors of each other; in the case of our airport example, they may even include carriers from nations in conflict, such as El Al and Iran Air. There may thus be value in not just protecting the network from misbehaviour by customers, but in protecting customers from each other.

The definition of 'protection' may be subtle. Customers can protect the confidentiality and integrity of their data using encryption, whether using VPNs or higher-layer mechanisms; the aspect of protection that falls to the network service provider is mainly resilience against denial-of-service (whether deliberate or inadvertent) though there may be some additional requirements in areas such as finance; Chinese-wall regulations may require a bank to separate different activities, while divisions involved in trading may want latency guarantees.

2. INFRASTRUCTURE SECURITY

The threat model for hardware and physical security is thus changing: growing scale makes physical attacks impossible to prevent completely. Janitors in a large airport can access many switch closets, shared facilities see many tenants having access to others' racks, and state actors can access devices in many countries.

Our threat model must assume physical compromise of devices, along with associated attacks that can be done by having physical access. We assume that some devices (the ones 'in the field') are unsafe. Once an attacker gains physical access to a device, he can open, modify, remove, or replace it. It is reasonable to say a number of switches are compromised at a given time, and sometimes controllers at the bottom of the hierarchy are compromised too (since they are deployed near switches). We also assume the communication channels connecting an unsecured switch to be insecure: just like the devices, the wires are also subjected to the same attacks. If an attacker can get access to a device, he also has access to its cabling.

Imagine an ISP needing to change a switch at a customer's site by ordering the equipment from a contractor. There is no guarantee of security at any of the steps from factory and shipping company to contractors and subcontractors and eventually to the customer. Even the customer may modify the equipment if there is incentive to do so. It is not just a matter of occasional access by secret policemen at points of presence in less well-governed countries.

In a time of conflict, the opportunity would exist for an opponent who had taken over one of the routers to use it to cause general havoc, by inserting false rules, or removing and replacing legitimate ones, and thus generally disrupting the switching fabric. If your network has 1,000 routers, and any one of them can disrupt it, as is the case today, then if only a few percent of your routers are compromised, you are vulnerable.

2.1 The opportunity

Our basic position in this paper is that the move from peer routers and switches, any of which can cause equal havoc if compromised, to a two-tier system of switches and controllers, gives a welcome opportunity to arrange things so that the compromise of a handful of switches will do no more than local damage. The controllers, where the intelligence resides, can mostly be moved out of harm's way into more protected environments. This enables us to protect the infrastructure better. What's more, programmable controllers mean opportunities for further and stronger separation via virtualisation, and the prospect of installing much more sophisticated security applications such as filtering, firewall and botnet countermeasures. This enables us to provide better protection functionality.

As for the infrastructure, a classical network-setup adapted to SDN, as illustrated in Figure 1 and currently deployed in data centres, might have a bottom layer of 1,000 switches, with each ten switches driven by a level 2 controller, every ten level 2 controllers driven by a level 1 device, and the ten level 1 devices coordinated by a master controller. Thus if we can arrange things so that only controllers can cause widespread outages if compromised, the number of critical components is reduced by a factor of ten. If we can further arrange things so that the compromise of a level 2 controller does little damage outside of its immediate neighbourhood, then we have reduced the number of points of serious failure by another order of magnitude.

Although Figure 1 illustrates an SDN hierarchy informed by datacenter practices, without much imagination it is plausible to map the components to those of an ISP (Network Operation Centre, Regional Offices, PoPs, etc.) and to the components of our airport example (where there are some central facilities, some in separate buildings and some on different floors of those buildings, connected in a hierarchy).

3. SOFTWARE SECURITY

Software plays a major role in an SDN, as the name implies. SDN, with its abstractions, enable operators to better manage their software and its security. However, programmability creates temptation, and complexity expands to fill the available space. Dependability will emerge from a number of factors including architecture, incentives and assurance.

3.1 Likely architecture

A typical SDN will have two or more layers of 'middle management' controllers between its root controller and the switch fabric. This is where the 'work' will be done, of creating virtual networks and supporting virtual services. For example, a new bank in example (3), or a new airline in example (4) above, will go to an infrastructure network manager and request a private network with a particular set of features; if approved, the manager will issue the necessary rules which will cascade down through controllers at levels 0, 1 and 2. The actual virtualisation will happen in the middle layers, at layers 1 and 2, with level-2 controllers issuing the necessary rules to the switches they control.

We assume the level 0 and level 1 controllers to be trusted, although with potential accidental configuration errors; and that there may be occasional compromises at level 2. However, there may be network application code (the SD applications) running on level 1 and level 2 controllers, which might misbehave, intentionally or not. As we noted, some proportion of the switches may be compromised at any one time; and, as the data packets being dealt with can come from anywhere, nothing is assumed of them.

We imagine that in time there will be many SDN applications that operators can choose to deploy. This will bring the same problems seen with application markets for mobile phones. Will we take the 'walled garden' approach of the iPhone, with some central authority that vets applications and developers, or the somewhat more freewheeling approach of Android, where all can play but applications are

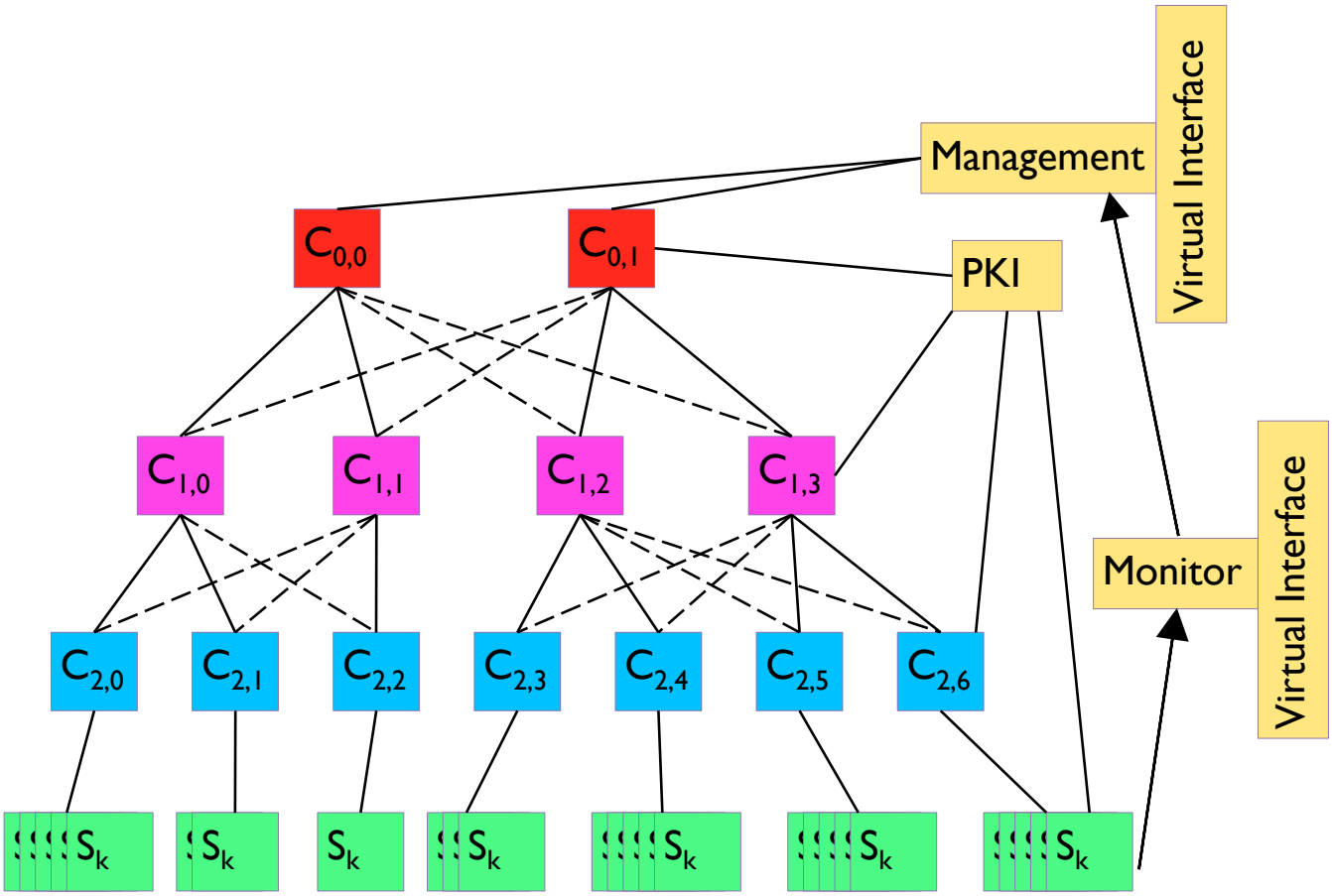

Figure 1: An SDN setup with hierarchical controllers and switches. Solid lines denote connections, and dotted lines backup connections. Note that the PKI, management, and monitoring services are conceptually drawn, and may not be physically separate from the main hierarchy.

removed from the play store once they are considered harmful? Many applications will contain too much code to verify, and even if their developers are honest and competent, they may still face commercial incentives to collect as much information as possible, or to give higher priority to their own traffic at the expense of their competitors'. Network engineers deciding how much access to grant an app may be more sophisticated than the typical Android user trying to decide whether a social networking app that asks for the ability to send text messages is exploitative — but the difficulties encountered with the manifests for phone apps bear careful thought. How should we design the set of permissions that will define and constrain the behaviour of an SDN app? How should the access control policies look like?

We believe this is an area that needs substantial and urgent research.

3.2 The OODA loop

The OODA loop (for Observe, Orient, Decide, and Act) was developed by air force analyst Colonel John Boyd to understand the factors leading to success and failure in combat. It has since been applied to information security management [1] and is a useful way of thinking about the management of software attacks. First we have to observe that a system is not behaving as it ought to, and in the case of a network that means having robust network monitoring mechanisms that are difficult for an attacker to subvert. We conceptualise these as a separate system, alongside the SDN core hierarchy, and reporting to the the top-level controllers and management servers, since that is where policies are specified.

The network monitoring mechanism may involve sampling a proportion of packets and verifying that the rules set by controllers at different levels are being complied with. This is another area in which significant research is needed; just as we need to translate a routing rule at level 0 or level

1 into forwarding rules that are passed by level 2 controllers to switches, so too do we need to translate such rules into sampling instructions to verify compliance for the packet streams. Other possible ways to enforce rules include using formal methods, conflict resolution, and proof of rule instalment.

A further problem with monitoring is that if there are virtual networks at different security levels — say for example an unclassified network in an airport used by a foreign airline, and a CONFIDENTIAL network used by the immigration service — then we do not want the management node of the first network to get packet monitoring data for the second. So the monitoring system has to provide strong separation of data on different virtual networks, if it is to provide feedback to virtual network managers directly.

The last frontier is the reaction system. Once an attack is detected, be it the presence of unauthorised software, anomalous flows, or a part of the network being down or partitioned, the SDN needs automatic procedures to recover. This may involve reconfiguration, rerouting, software reinstallation and other actions, even prior to the invocation of human direction.

Mechanisms that can assist in recovery from attack include a trustworthy core network. We argue that SDN can, by locating much of the trust in higher-level controllers, limit the damage that can be done by router compromise. But these controllers had better be able to talk to each other. Since the trusted controllers are few in numbers and therefore easier to physically protect, an attacker trying to partition the network will have a hard time to partition the core, or separate the core from a chunk of the periphery.

Another relevant mechanism is trusted boot. At present, recovery from attack might involve trusted upgrade of router firmware followed by rebooting all the devices and waiting for the network to come up and re-converge. If some routers are under enemy control and participate maliciously in this process, they might frustrate it. Again, an advantage of SDN is that one can bring up the network from the trusted core outwards; hierarchical reboot should be both faster and less open to disruption than a traditional spanning-tree approach.

3.3 Network Applications

Currently imagined SDN network security applications would be a naïve transformation of packet-filtering and firewall rules into network-wide mechanisms. However, while a natural transformation is possible, and using SDN as a simpler system interface has motivated some to unify once disparate network-control systems [2], SDN opportunities are more than a unified API to the switching domain.

For example, the abstraction implicit in SDN permits the network to be considered as a graph not merely the outcome of routing protocols; this permits a formalism to take hold. By allowing the SDN to separate the desired outcome from the implementation specifics; firewall rules, packet-filtering and access controls can be transformed into explicit permission control: e.g., explicit routability [3]. The rules may then permit strongly reasoned properties, thus leading to a network about which correctness (and incorrectness) may also be reasoned. From a more practical standpoint, such abstractions allow operators to work with the network as a whole, rather than going deeper and work with individual switches. This shift is a good thing.

Collecting the forwarding rules and requests from switches to a single location also permits an unrealised level of control. While scalability is not immediately clear, we now have all of the warranted and unwarranted traffic data within the network. The potentials of using this for auditing and information flow analysis is immense. Among others, SDN makes available an interesting potential for tackling botnet outbreaks as well as adapting and reacting to other forms of network attacks [4, 5].

4. PROTOCOL SECURITY

At present, SSL/TLS is optional in OpenFlow but not widely used or implemented. This is natural enough in the controlled environment of a data centre, but is already unsatisfactory for a large Autonomous System (AS). The difficulties of key management are already known: keys for SSH and SSL/TLS are often managed by supplying devices with key material on USB devices. Things will become more complex with the coming deployment of BGPSEC [6].

The problem here is to work out what can be reused and what is different in an SDN setup. Many key-related operations can be reused with little or no adaptation. The primary challenge is bootstrapping: how do you bootstrap a key into a device in an operationally feasible way, particularly when the hands doing the work are not always trusted? And how do you bootstrap a network of devices while using the same devices for network connectivity?

Since multiple protocols are followed by multiple parties, one possible concern is maintaining consistency of device identifiers. This is an authentication problem at one level, but at another it is about maintaining a consistent naming scheme that operates across multiple protocols. It may make strategic sense to use a single reliable authentication system to bootstrap the others.

The design of this may require some care. Consider for example the 'power failure' scenario. Adding devices incrementally may be easy, especially with some human help, but how do you bring up hundreds or thousands of devices all at once, if this involves doing authentication with TLS or Kerberos? Invoking a human operator is completely infeasible at this scale; the network needs to do it automatically. So how is trust established during a network restart, and how does it converge in a reasonable time? Here too the hierarchical structure of a typical SDN may give a significant advantage at large scale.

4.1 Factors to Consider

Key-related operations carry a cost, and we need to con-

sider how this cost scales for key setup, update, storage, and revocation. These are not new problems; there was significant discussion in the late 1990s about the relative cost of symmetric and asymmetric cryptography. SDN holds out the prospect of deployments are of sufficient scale for such considerations to matter.

Another consideration is heterogeneity. Rarely do operators upgrade the whole network all at once. When SDN is deployed, operators might start from the edges to ensure their core network will not be affected if something goes wrong. If one edge of the network needs to speak SDN to another edge, they will traverse through a traditional network. How should this communication channel be handled? IPv6 in a primarily IPv4 world has faced with the same problem, perhaps lessons can be learned from IPv6 deployments.

There are also issues of mating authentication in SDN with that of DNSSEC, and BGPSEC once it is deployed. It is very likely that these will co-exist in some way, either directly leveraging each other or one embedding another. Cross system authentication issues need to be considered.

5. MANAGEMENT INTERFACE

Finally, the management interface will be critical. Many of the significant problems with current networks are down to usability; the management interface we have now is either CLI (which are decades old) or a collection of legacy scripts built and carefully balanced on top of the CLI. Operators have a high level view of what they want their network to do, but are then forced to implement that using very low level kludges. This leads to misconfigurations, route leaks and worse. Things are made worse by the fact that the properties/capabilities of each router blade are different, so the semantics are all over the place and a delicate balance between what you want the device to do and what it can do (or be made to do) 'well enough' has to be struck -- often by human beings, who make errors.

In short, the biggest threat for years has been operator error. We have seen with BGP networks many instances of routing incidents caused by fat-fingering, where an operator mistakenly inputs wrong commands into the CLI (Youtube-Pakistan in 2008, China Telecom in 2010).

SDN can improve this. The goal is to configure the network, not the routers. By having a hierarchy of controllers, with a management interface at the top, each layer can (in theory at least) speak the right level of abstraction for that layer. This is the way to make things more usable, and to enable networks to scale up. In the operators community there is a consensus that router command interfaces need improvement; one of the underlying causes is vendor lock-in, as the major vendors have, either for commercial necessity or product differentiation, made their CLIs subtly different. SDN can break the lock-in and create the chance to build more usable network management tools.

SDN can abstract this problem away. OpenFlow, the protocol to realise SDN with the most momentum, is heading in the right direction. It specifies a standard set of commands with which a controller can interact with an OpenFlow-capable switch. However, the controller-controller and controller-application interfaces are yet to be defined. And it goes without saying that any automated configuration system or command-translation software can have its own bugs and vulnerabilities. It is also essential to verify the correctness of the output of such helper software.

6. RELATED WORK

The idea of SDNs regained real traction after the introduction of OpenFlow [7]. Since then, numerous works have looked at different aspects of OpenFlow and SDN in general. There are three directions most relevant to our work: abstractions, controllers, and monitoring.

On the abstractions front, we have seen language-based methods used in enforcement of policies, security goals, and network operations. There is FSL [8] which is a policy language that lets operators write policies in a enforceable way. Another work uses a programming abstraction to define and isolate network slices [9]. Frenetic is also an ongoing project that aims to bring an abstraction to network programming, especially in querying for state, defining policies, and updating network configurations [10]. In updating network configurations, we have also seen a work that abstracts away potential inconsistencies during a network-wide configuration update [11].

Controllers have also been a focus point for research. We have seen FRESCO, a security application development framework to help detect threats and mitigate them in an OpenFlow application [5]. FortNOX is another work that further brings security to OpenFlow by extending NOX to enforce and detect policy contradictions in real time [4]. In each of these cases the OpenFlow single point of control is central to the application. In contrast, the number of controllers, how they scale, and their performance implications are studied in [12].

There is also some work in monitoring and interactions with the data plane. We see an effort to dynamically analyze OpenFlow application code and to identify bugs using symbolic execution in [13], with an earlier work that applies model-checking on the state space of the SDN system [14]. VeriFlow sits between controllers and forwarding engines to check for violations in network-wide invariants, as the rules are inserted [15]. FlowSense takes a different approach, and lets switches push network performance numbers upwards to the controller, in order to monitor and measure the network [16].

7. CONCLUSIONS

Software defined networks are currently seen as a means of reducing equipment and operating costs, freeing operators from the limitations of proprietary router architectures, and providing uniform, simplified and closer control of multiple platforms.

In this position paper, we argue that SDNs have *another* string to their bow. As networks scale up, it is increasingly likely that some routers will be under hostile control. With existing architectures and protocols, this enables them to be used for extremely disruptive denial-of-service attacks leading to large scale havoc. SDN offers a more hierarchical approach to network management in which failures are easier to spot and more able to be locally contained. The abstraction that SDN brings allow for conceptually centralised view of the network both in control and data planes, letting us continuously monitor the network for unexpected behaviour. They can also enable tighter management of upgrades and faster recovery in the event of disruption: accidental or malicious. Realising these benefits will involve some interesting security research.

8. ACKNOWLEDGMENTS

We would like to thank Peter Neumann, Phillip Porras, Vinod Yegneswaran, Anil Madhavapeddy, and Charalampos Rotsos for helpful discussions in the early stage of this work. We would also like to thank the audience at the Security Protocols Workshop '13 where some of the earlier ideas were presented.

This work was sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL), in part under contract FA8750-11-C-0249, and in part under contract FA8750-13-2-0023. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views, opinions, and/or findings contained in this report are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency, the Air Force Research Laboratory or the Department of Defense.

9. REFERENCES

[1] S. Clark, M. Blaze, and J. M. Smith, "The casino and the OODA loop," in *Security Protocols XX*, ser. Lecture Notes in Computer Science, B. Christianson, J. Malcolm, F. Stajano, and J. Anderson, Eds. Springer Berlin Heidelberg, 2012, vol. 7622, pp. 60–63.

[2] U. Hoelzle, "OpenFlow @ Google," 2012, keynote address at the Open Network Summit.

[3] T. Roscoe, S. Hand, R. Isaacs, R. Mortier, and P. Jardetzky, "Predicate routing: enabling controlled networking," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 65–70, Jan. 2003.

[4] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu, "A security enforcement kernel for OpenFlow networks," in *Proceedings of the first workshop on Hot topics in software defined networks*, ser. HotSDN '12. ACM, 2012, pp. 121–126.

[5] S. Shin, P. Porras, V. Yegneswaran, M. Fong, G. Gu, and M. Tyson, "Fresco: Modular composable security services for software-defined networks," *Internet Society NDSS (Feb. 2013). To appear*, 2013.

[6] M. Lepinski (Ed.), "BGPSEC Protocol Specification," Internet Engineering Task Force, Feb. 2013. [Online]. Available: http://www.ietf.org/id/draft-ietf-sidr-bgpsec-protocol-07.txt

[7] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[8] T. Hinrichs, N. Gude, M. Casado, J. Mitchell, and S. Shenker, "Expressing and enforcing flow-based network security policies," University of Chicago Technical Report, Tech. Rep., 2008.

[9] S. Gutz, A. Story, C. Schlesinger, and N. Foster, "Splendid isolation: a slice abstraction for software-defined networks," in *Proceedings of the first workshop on Hot topics in software defined networks*, ser. HotSDN '12. ACM, 2012, pp. 79–84.

[10] N. Foster, A. Guha, M. Reitblatt, A. Story, M. J. Freedman, N. P. Katta, C. Monsanto, J. Reich, J. Rexford, C. Schlesinger, A. Story, and D. Walker, "Languages for software-defined networks," *Communications Magazine, IEEE*, vol. 51, no. 2, pp. 128–134, 2013.

[11] M. Reitblatt, N. Foster, J. Rexford, C. Schlesinger, and D. Walker, "Abstractions for network update," in *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, ser. SIGCOMM '12. ACM, 2012, pp. 323–334.

[12] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proceedings of the first workshop on Hot topics in software defined networks*, ser. HotSDN '12. ACM, 2012, pp. 7–12.

[13] M. Canini, D. Kostic, J. Rexford, and D. Venzano, "Automating the testing of openflow applications," in *The 1st International Workshop on Rigorous Protocol Engineering (WRiPE)*, 2011.

[14] M. Canini, D. Venzano, P. Peresini, D. Kostic, and J. Rexford, "A NICE way to test openflow applications," *NSDI, Apr*, 2012.

[15] A. Khurshid, W. Zhou, M. Caesar, and P. Godfrey, "VeriFlow: verifying network-wide invariants in real time," in *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012, pp. 49–54.

[16] C. Yu, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, and H. V. Madhyastha, "FlowSense: Monitoring network utilization with zero measurement cost," in *Proceedings of Passive and Active Measurement Conference (PAM)*, 2013.